Package 'RSKC'

July 2, 2014

Type Package

Title Robust sparse K-means

Version 2.4.1

Date 2011-09-11

Author Yumi Kondo

Maintainer Yumi Kondo <y.kondo@stat.ubc.ca>

Description This package contains a function RSKC which runs the robust sparse K-means clustering algorithm.

License GPL (>= 2)

LazyLoad yes

LazyData yes

Repository CRAN

Depends flexclust, stats, R (>= 2.14.0)

Imports

Date/Publication 2014-06-15 07:33:29

NeedsCompilation yes

R topics documented:

CER *Classification Error Rate (CER)*

Description

Compute the classification error rate of two partitions.

Usage

```
CER(ind, true.ind,nob=length(ind))
```

Arguments

ind	Vector, containing the cluster labels of each case of a partition 1.
true.ind	Vector, containing the cluster labels of each case of a partition 2.
nob	The number of cases (the length of the vector ind and true ind)

Value

Return a CER value. CER = 0 means perfect agreement between two partitions and CER = 1 means complete disagreement of two partitions. Note: 0 <= CER <= 1

Note

This function uses comb, which generates all combinations of the elements in the vector ind. For this reason, the function CER is not suitable for vector in a large dimension.

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

References

H. Chipman and R. Tibshirani. Hybrid hierarchical clustering with applications to microarray data. Biostatistics, 7(2):286-301, 2005.

Examples

```
vec1<-c(1,1,1,2,3,3,3,2,2)
vec2<-c(3,3,3,1,1,2,2,1,1)
CER(vec1,vec2)
```

`Clest` *An implementation of Clest with robust sparse K-means. CER is used as a similarity measure.*

Description

The function `Clest` performs Clest (Dudoit and Fridlyand (2002)) with CER as the measure of the agreement between two partitions (in each training set). The following clustering algorithm can be used: K-means, trimmed K-means, sparse K-means and robust sparse K-means.

Usage

```
Clest(d, maxK, alpha, B = 15, B0 = 5, nstart = 1000,

      L1 = 6, beta = 0.1, pca = TRUE, silent=FALSE)
```

Arguments

`d`	A numerical data matrix (`N` by `p`) where `N` is the number of cases and `p` is the number of features. The cases are clustered.
`maxK`	The maximum number of clusters that you suspect.
`alpha`	See `RSKC`.
`B`	The number of times that an observed dataset `d` is randomly partitioned into a learning set and a training set. Note that each generated reference dataset is partitioned into a learning and a testing set only once to ease the computational cost.
`B0`	The number of times that the reference dataset is generated.
`nstart`	The number of random initial sets of cluster centers at Step(a) of robust sparse K-means clustering.
`L1`	See `RSKC`.
`beta`	0 <= `beta` <= 1: significance level. Clest chooses the number of clusters that returns the strongest significant evidence against the hypothesis H0 : K = 1.
`pca`	Logical, if `TRUE`, then reference datasets are generated from a PCA reference distribution. If `FALSE`, then the reference data set is generated from a simple reference distribution.
`silent`	Logical, if `TRUE`, then the number of iteration on progress is not printed.

Value

`K`	The solution of Clest; the estimated number of clusters.
`result.table`	A real matrix (`maxK-1` by 4). Each row represents `K=2,...,maxK` and columns represent the test statistics (=observed CER-reference CER), observed CER, reference CER and P-value.

referenceCERs	A matrix (B0 by maxK-1), containing CERs of testing datasets from generated datasets for each K=2,...,maxK.
observedCERs	A matrix (B by maxK-1), containing CERs of B testing sets for each K=2,...,maxK.
call	The matched call.

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

References

Yumi Kondo (2011), Robustificaiton of the sparse K-means clustering algorithm, MSc. Thesis, University of British Columbia http://hdl.handle.net/2429/37093

S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. Genome Biology, 3(7), 2002.

Examples

```
## Not run:
# little simulation function
sim <-
function(mu,f){
   D<-matrix(rnorm(60*f),60,f)
   D[1:20,1:50]<-D[1:20,1:50]+mu
   D[21:40,1:50]<-D[21:40,1:50]-mu
   return(D)
   }

 set.seed(1)
 d<-sim(1.5,100); # non contaminated dataset with noise variables

# Clest with robust sparse K-means
rsk<-Clest(d,5,alpha=1/20,B=3,B0=10, beta = 0.05, nstart=100,pca=TRUE,L1=3,silent=TRUE);
# Clest with K-means
k<-Clest(d,5,alpha=0,B=3,B0=10, beta = 0.05, nstart=100,pca=TRUE,L1=NULL,silent=TRUE);

## End(Not run)
```

DBWorld *E-mails from DBWorld mailing list*

Description

The dataset contains n= 64 bodies of e-mails in binary bag-of-words representation which Filannino manually collected from DBWorld mailing list.

DBWorld mailing list announces conferences, jobs, books, software and grants.

Filannino applied supervised learning algorithm to classify e-mails between “announces of conferences” and “everything else”.

Out of 64 e-mails, 29 are about conference announcements and 35 are not.

Every e-mail is represented as a vector containing p binary values, where p is the size of the vocabulary extracted from the entire corpus with some constraints: the common words such as "the", "is" or "which", so-called stop words, and words that have less than 3 characters or more than 30 chracters are removed from the dataset.

The entry of the vector is 1 if the corresponding word belongs to the e-mail and 0 otherwise.

The number of unique words in the dataset is p=4702.

The dataset is originally from the UCI Machine Learning Repository DBWorldData.

`rawDBWorld` is a list of 64 objects containing the original E-mails.

Usage

```
data(DBWorld)
data(rawDBWorld)
```

Details

See Bache K, Lichman M (2013). for details of the data descriptions. The original dataset is freely available from USIMachine Learning Repository website http://archive.ics.uci.edu/ml/datasets/DBWorld+e-mails

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

References

Bache K, Lichman M (2013). UCI Machine Learning Repository." http://archive.ics.uci.edu/ml/datasets

Filannino, M., (2011). 'DBWorld e-mail classification using a very small corpus', Project of Machine Learning course, University of Manchester.

Examples

```
## Not run:
data(DBWorld)
data(rawDBWorld)

## End(Not run)
```

DutchUtility *Multiple Features Data Set of Robert P.W. Duin.*

Description

This dataset consists of features of handwritten numerals ('0'–'9') (K=10) extracted from a collection of Dutch utility maps.

Two hundred patterns per class (for a total of 2,000 (=N) patterns) have been digitized in binary images.

Raw observations are 32x45 bitmmaps, which are divided into nooverlapping blocks of 2x3 and the number of pixels are counted in each block.

This generate p=240 (16x15) variable, recodring the normalized counts of pixels in each block and each element is an integer in the range 0 to 6.

`rownames` of `DutchUtility` contains the true digits and `colnames` of it contains the position of the block matrix, from which the normalized counts of pixels are taken.

Usage

```
data(DutchUtility)
showDigit(index,cex.main=1)
```

Arguments

`index`	A scalar containing integers between 1 and 2000. The function `ShowDigit` regenerates the sampled versions of the original images may be obtained (15x16 pixels). (the source image (32x45) dataset is lost)
`cex.main`	Specify the size of the title text with a numeric value of length 1.

Details

The original dataset is freely available from USIMachine Learning Repository (Frank and Asuncion (2010)) website http://archive .ics.uci.edu/ml/datasets.html.

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

References

Frank A, Asuncion A (2010). UCI Machine Learning Repository." http://archive.ics.uci.edu/ml.

Examples

```
## Not run:

data(DutchUtility)

truedigit <- rownames(DutchUtility)
(re <- RSKC(DutchUtility,ncl=10,alpha=0.1,L1=5.7,nstart=1000))
Sensitivity(re$labels,truedigit)
table(re$labels,truedigit)

## Check the bitmap of the trimmed observations
showDigit(re$oW[1])
## Check the features which receive zero weights
names(which(re$weights==0))

## End(Not run)
```

`optd` *Optical Recognition of Handwritten Digits of Frank A, Asuncion A (2010).*

Description

The dataset describes n = 1797 digits from 0 to 9 (K = 10), handwritten by 13 subjects. Raw observations are 32x32 bitmaps, which are divided into nonoverlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates p = 64 (= 8x8) variable, recording the normalized counts of pixels in each block and each element is an integer in the range 0 to 16. The row names of the matrix optd contains the true labels (between 0 and 9), and the column names of it contains the position of the block in original bitmap.

Usage

```
data(optd)
showbitmap(index)
```

Arguments

`index`	A vector containing integers between 1 and 1797. Given the observation indices, the `showbitmap` returns their original 32 by 32 bitmaps on R console.

Details

The original dataset is freely available from USIMachine Learning Repository (Frank and Asuncion (2010)) website http://archive .ics.uci.edu/ml/datasets.html.

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

References

Frank A, Asuncion A (2010). UCI Machine Learning Repository." http://archive.ics.uci.edu/ml.

Examples

```
## Not run:

data(optd)

truedigit <- rownames(optd)
(re <- RSKC(optd,ncl=10,alpha=0.1,L1=5.7,nstart=1000))
Sensitivity(re$labels,truedigit)
table(re$labels,truedigit)

## Check the bitmap of the trimmed observations
showbitmap(re$oW)
## Check the features which receive zero weights
names(which(re$weights==0))

## End(Not run)
```

`revisedsil` *The revised silhouette*

Description

This function returns a revised silhouette plot, cluster centers in weighted squared Euclidean distances and a matrix containing the weighted squared Euclidean distances between cases and each cluster center. Missing values are adjusted.

Usage

```
revisedsil(d,reRSKC=NULL,CASEofINT=NULL,col1="black",
CASEofINT2 = NULL, col2="red", print.plot=TRUE,
W=NULL,C=NULL,out=NULL)
```

Arguments

`d`	A numerical data matrix, `N` by `p`, where `N` is the number of cases and `p` is the number of features.
`reRSKC`	A list output from RSKC function.
`CASEofINT`	Necessary if print.plot=TRUE. A vector of the case indices that appear in the revised silhouette plot. The revised silhouette widths of these indices are colored in `col1` if `CASEofINT != NULL`. The average silhouette of each cluster printed in the plot is computed EXCLUDING these cases.
`col1`	See `CASEofINT`.

CASEofINT2	A vector of the case indices that appear in the revised silhouette plot. The indices are colored in col2.
col2	See CASEofINT2
print.plot	If TRUE, the revised silhouette is plotted.
W	Necessary if reRSKC = NULL. A positive real vector of weights of length p.
C	Necessary if reRSKC = NULL. An integer vector of class labels of length N.
out	Necessary if reRSKC = NULL. Vector of the case indices that should be excluded in the calculation of cluster centers. In RSKC, cluster centers are calculated without the cases that have the furthest 100*alpha % Weighted squared Euclidean distances to their closest cluster centers. If one wants to obtain the cluster centers from RSKC output, set out = <RSKCoutput>$oW.

Value

trans.mu	Cluster centers in reduced weighted dimension. See example for more detail.
WdisC	N by ncl matrix, where ncl is the prespecified number of clusters. It contains the weighted distance between each case and all cluster centers. See example for more detail.
sil.order	Silhouette values of each case in the order of the case index.
sil.i	Silhouette values of cases ranked by decreasing order within clusters. The corresponding case index are in obs.i

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

References

Yumi Kondo (2011), Robustificaiton of the sparse K-means clustering algorithm, MSc. Thesis, University of British Columbia http://hdl.handle.net/2429/37093

Examples

```
# little simulation function
sim <-
function(mu,f){
   D<-matrix(rnorm(60*f),60,f)
   D[1:20,1:50]<-D[1:20,1:50]+mu
   D[21:40,1:50]<-D[21:40,1:50]-mu
   return(D)
   }


### output trans.mu ###

p<-200;ncl<-3
# simulate a 60 by p data matrix with 3 classes
```

```
d<-sim(2,p)
# run RSKC
re<-RSKC(d,ncl,L1=2,alpha=0.05)
# cluster centers in weighted squared Euclidean distances by function sil
sil.mu<-revisedsil(d,W=re$weights,C=re$labels,out=re$oW,print.plot=FALSE)$trans.mu
# calculation
trans.d<-sweep(d[,re$weights!=0],2,sqrt(re$weights[re$weights!=0]),FUN="*")
class<-re$labels;class[re$oW]<-ncl+1
MEANs<-matrix(NA,ncl,ncol(trans.d))
for ( i in 1 : 3) MEANs[i,]<-colMeans(trans.d[class==i,,drop=FALSE])
sil.mu==MEANs
# coincides

### output WdisC ###

p<-200;ncl<-3;N<-60
# generate 60 by p data matrix with 3 classes
d<-sim(2,p)
# run RSKC
re<-RSKC(d,ncl,L1=2,alpha=0.05)
si<-revisedsil(d,W=re$weights,C=re$labels,out=re$oW,print.plot=FALSE)
si.mu<-si$trans.mu
si.wdisc<-si$WdisC
trans.d<-sweep(d[,re$weights!=0],2,sqrt(re$weights[re$weights!=0]),FUN="*")
WdisC<-matrix(NA,N,ncl)
for ( i in 1 : ncl) WdisC[,i]<-rowSums(scale(trans.d,center=si.mu[i,],scale=FALSE)^2)
# WdisC and si.wdisc coincides
```

RSKC *Robust Sparse K-means*

Description

The robust sparse *K*-means clustering method by Kondo (2011). In this algorithm, sparse *K*-means (Witten and Tibshirani (2010)) is robustified by iteratively trimming the prespecified proportion of cases in the weighted squared Euclidean distances and the squared Euclidean distances.

Usage

```
RSKC(d, ncl, alpha, L1 = 12, nstart = 200,
silent=TRUE, scaling = FALSE, correlation = FALSE)
```

Arguments

d	A numeric matrix of data, N by p, where N is the number of cases and p is the number of features. Cases are partitioned into ncl clusters. Missing values are accepted.
ncl	The prespecified number of clusters.

alpha	0 <= alpha <= 1, the proportion of the cases to be trimmed in robust sparse K-means. If alpha > 0 and L1 >= 1 then RSKC performs robust sparse K-means. If alpha > 0 and L1 = NULL then RSKC performs trimmed K-means. If alpha = 0 and L1 >=1 then RSKC performs sparse K-means (with the algorithm of Lloyd (1982)). If alpha = 0 and L1 = NULL then RSKC performs K-means (with the algorithm of Lloyd). For more details on trimmed K-means, see Gordaliza (1991a), Gordaliza (1991b).
L1	A single L1 bound on weights (the feature weights). If L1 is small, then few features will have non-zero weights. If L1 is large then all features will have non-zero weights. If L1 = NULL then RSKC performs nonsparse clustering (see alpha).
nstart	The number of random initial sets of cluster centers in every step (a) which performs K-means or trimmed K-means.
silent	If TRUE, then the processing step is not printed.
scaling	If TRUE, RSKC subtracts the each entry of data matrix by the corresponding column mean and divide it by the corresponding column SD: see scale
correlation	If TRUE, RSKC centers and scales the rows of data before the clustering is performed. i.e., trans.d = t(scale(t(d))) The squared Euclidean distance between cases in the transformed dataset trans.d is proportional to the dissimilality measure based on the correlation between the cases in the dataset d

Details

Robust sparse K-means is a clustering method that extends the sparse K-means clustering of Witten and Tibshirani to make it resistant to oultiers by trimming a fixed proportion of observations in each iteration.

These outliers are flagged both in terms of their weighted and unweighted distances to eliminate the effects of outliers in the selection of feature weights and the selection of a partition.

In Step (a) of sparse K-means, given fixed weights, the algorithm aims to maximize the objective function over a partition i.e. it performs K-means on a weighted dataset. Robust sparse K-means robustifies Step (a) of sparse K-means by performing trimmed K-means on a weighted dataset: it trims cases in weighted squared Euclidean distances.

Before Step (b), where, given a partition, the algorithm aims to maximize objective function over weights, the robust sparse K-means has an intermediate robustifying step, Step (a-2). At this step, it trims cases in squared Euclidean distances.

Given a partition and trimmed cases from Step (a) and Step (a-2), the objective function is maximized over weights at Step(b). The objective function is calculated without the trimmed cases in Step (a) and Step(a-2).

The robust sparse K-means algorithm repeat Step (a), Step (a-2) and Step (b) until a stopping criterion is satisfied.

For the calculation of cluster centers in the weighted distances, see revisedsil.

Value

N	The number of cases.
p	The number of features.
ncl	See ncl above.
L1	See L1 above.
nstart	See nstart above.
alpha	See alpha above.
scaling	See scaling above.
correlation	See correlation above.
missing	It is TRUE if at least one point is missing in the data matrix, d.
labels	An integer vector of length N, set of cluster labels for each case. Note that trimmed cases also receive the cluster labels.
weights	A positive real vector of length p, containing weights on each feature.
WBSS	A real vector containing the weighted between sum of squares at each Step (b). The weighted between sum of squares is the objective function to maximize, excluding the prespecified proportions of cases. The length of this vector is the number of times that the algorithm iterates the process steps (a),(a-2) and (b) before the stopping criterion is satisfied. This is returned only if L1 is numeric and > 1.
WWSS	A real number, the within cluster sum of squares at a local minimum. This is the objective function to minimize in nonsparse methods. For robust clustering methods, this quantity is calculated without the prespecified proportions of cases. This is returned only if L1=NULL,
oE	Indices of the cases trimmed in squared Euclidean distances.
oW	Indices of the cases trimmed in weighted squared Euclidean distances. If L1 =NULL, then oW are the cases trimmed in the Euclidean distance, because all the features have the same weights, i.e., 1's.

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

References

A. Gordaliza. Best approximations to random variables based on trimming procedures. Journal of Approximation Theory, 64, 1991a.

A. Gordaliza. On the breakdown point of multivariate location estimators based on trimming procedures. Statistics & Probability Letters, 11, 1991b.

Y. Kondo (2011), Robustificaiton of the sparse K-means clustering algorithm, MSc. Thesis, University of British Columbia http://hdl.handle.net/2429/37093

D. M. Witten and R. Tibshirani. A framework for feature selection in clustering. Journal of the American Statistical Association, 105(490) 713-726, 2010.

S.P. Least Squares quantization in PCM. IEEE Transactions on information theory, 28(2): 129-136, 1982.

Examples

```
# little simulation function
sim <-
function(mu,f){
   D<-matrix(rnorm(60*f),60,f)
   D[1:20,1:50]<-D[1:20,1:50]+mu
   D[21:40,1:50]<-D[21:40,1:50]-mu
   return(D)
   }

set.seed(1);d0<-sim(1,500)# generate a dataset
true<-rep(1:3,each=20) # vector of true cluster labels
d<-d0
ncl<-3
for ( i in 1 : 10){
   d[sample(1:60,1),sample(1:500,1)]<-rnorm(1,mean=0,sd=15)
}

# The generated dataset looks like this...
pairs(
      d[,c(1,2,3,200)],col=true,
      labels=c("clustering feature 1",
      "clustering feature 2","clustering feature 3",
      "noise feature1"),
      main="The sampling distribution of 60 cases colored by true cluster labels",
      lower.panel=NULL)


# Compare the performance of four algorithms
###3-means
r0<-kmeans(d,ncl,nstart=100)
CER(r0$cluster,true)

###Sparse 3-means
#This example requires sparcl package
#library(sparcl)
#r1<-KMeansSparseCluster(d,ncl,wbounds=6)
# Partition result
#CER(r1$Cs,true)
# The number of nonzero weights
#sum(!r1$ws<1e-3)

###Trimmed 3-means

r2<-RSKC(d,ncl,alpha=10/60,L1=NULL,nstart=200)
CER(r2$labels,true)

###Robust Sparse 3-means
r3<-RSKC(d,ncl,alpha=10/60,L1=6,nstart=200)
# Partition result
CER(r3$labels,true)
r3
```

```
### RSKC works with datasets containing missing values...
# add missing values to the dataset
set.seed(1)
for ( i in 1 : 100)
{
d[sample(1:60,1),sample(1,500,1)]<-NA
}
r4 <- RSKC(d,ncl,alpha=10/60,L1=6,nstart=200)
```

`Sensitivity` *Compute the sensitivities (probability of true positive) of each cluster*

Description

The sensitivity or conditional probability of the correct classification of cluster k is calculated as follows:

First, the proportions of observations whose true cluster label is k are computed for each classified clusters.

Then the largest proportion is selected as the conditional probability of the correct classification.

Since this calculation can return 1 for sensitivities of all clusters if all observations belong to one cluster, we also report the observed cluster labels returned by the algorithms.

Usage

```
Sensitivity(label1, label2)
```

Arguments

`label1`	A vector of length N, containing the cluster labels from any clustering algorithms.
`label2`	A vector of length N, containing the true cluster labels.

Author(s)

Yumi Kondo <y.kondo@stat.ubc.ca>

Examples

```
vec1<-c(1,1,1,2,3,3,3,2,2)
vec2<-c(3,3,3,1,1,2,2,1,1)
Sensitivity(vec1,vec2)
```

Index